

REMARKS

Claims 1–14, 25, and 26 were previously pending in this application.

Claims 1, 25 and 26 are amended. No claims are added. Claims 1–14, 25 and 26 remain pending.

35 U.S.C. § 102 Rejections

Claims 1–5, 7–9, 25 and 26

Claims 1–5, 7–9, 25 and 26 stand rejected under 35 U.S.C. 102(b) as being anticipated by Viroli and Natali ("Parametric Polymorphism in Java through the Homogeneous Translation LM: Gathering Type Descriptors at Load-Time", DEIS Technical Report No. DEIS-LIA-00-001, 2000) (hereinafter "Viroli"). Applicant respectfully traverses the rejection.

Generally, the presently claimed subject matter is concerned with providing a typing context for execution of operations that involve parametric polymorphism. The typing context for each polymorphic expression is characterized using a dynamically allocatable runtime type descriptor (RTD) that records the *exact* type of an associated generic type.

Type of Response: Final Response
Application Number: 10/025,270
Attorney Docket Number: 180610.01
Filing Date: 12/18/2001

Viroli describes a system extending past parametric polymorphism systems for Java such as Pizza, Generic Java (GJ), and NextGen with a homogeneous translation approach. The system of Viroli, in general, implements the translation for parameterized types in which the *erasure technique* is employed. That is, each parameterized type within a class is removed or *erased* and replaced with the Java super-type **Object**. When the parameterized type is instantiated, down-cast type conversions are performed to narrow **Object** to a correctly-typed (i.e. not exact) element (see Viroli, page 3, section 1 and page 8, section 3).

In contrast, the presently claimed subject matter does not include such a translation. Therefore, an "open type" class in the presently claimed subject matter may have reflection performed upon it at run time and a true representation of the "open-type" class may be returned, while in Viroli the translated open-type class will contain references to the Java super-type **Object** and Java reflection performed upon it at run time may return references to **Object** that will be "erased" when the "open-type" class is instantiated with a type.

Claim 1

Claim 1 has been amended and now recites a "computer program product encoding a computer program for executing on a computer system a computer process for dynamically generating typing context data associated with a typing-context-relevant-code-point being executed within a typing context in a dynamic execution environment." The computer process comprises steps of:

(1) "encountering the typing-context-relevant-code-point in the typing context during execution of the program;"

(2) "identifying a typing context handle associated with the typing context, the typing context handle referencing a typing context data structure associated with the typing context;"

(3) "computing the typing context data associated with the typing-context-relevant-code-point;"

(4) "dynamically allocating a field in the typing context data structure associated with the typing-context-relevant-code-point, the field describing the exact type of the typing-context-relevant-code-point in the typing context;"

and

Type of Response: Final Response
Application Number: 10/025,270
Attorney Docket Number: 180610.01
Filing Date: 12/18/2001

(5) "recording the typing context data in the field of the typing context data structure." (Amendment emphasized).

Claim 1 has been amended to clarify that the field in the context data structure associated with the typing-context-relevant-code-point describes the exact type of the typing-context-relevant-code-point in the typing context. See Specification, page 8, lines 17-19, "RTD-relevant-code-points are mapped into a typing context data structure ('TC data structure') to an RTD describing the exact type of the generic type in the current typing context."

In the Official Action of 8/25/2005, in rejecting claim 1 the Examiner suggests the "typing context data structure" of claim 1 is disclosed in Viroli at "page 12, section 4.2, first paragraph, which shows a type descriptor manager, type descriptors, and hash table." Viroli discloses that the type descriptor does not describe an exact type, rather, Viroli discloses that a type descriptor is a translated object that may store all the needed information about the type. For example, see Viroli, page 6, section 3, first paragraph, "each time a type operation over a given parametric type instantiation $A\langle T \rangle$ has to be realized, an object t storing all the needed information about the type $A\langle T \rangle$ is created, and the operation is translated into an appropriate method of t . Such objects will

Type of Response: Final Response
Application Number: 10/025,270
Attorney Docket Number: 180610.01
Filing Date: 12/18/2001

work as type descriptors.” Viroli further discloses that these type descriptors are those stored in the hash table by the type descriptor manager.

Therefore, if as the Examiner suggests, the typing context data structure of claim 1 is disclosed in Viroli as a type descriptor manager storing type descriptors in a hash table then Viroli further discloses that type descriptors describe translated objects and do not describe the exact type as in claim 1.

Such a distinction is illustrated in the specification at page 9, lines 10–16, “at the instantiation of an object of a generic class, the typing context of the object is recorded in a TC [type context] data structure accessible through the object itself (e.g. through the object’s pointer to its virtual table (vtable)). For example, the TC data structure may be appended to the vtable directly or through pointer indirection. In this manner, open–type expressions within the generic class of the object may determine the current typing context of the object and therefore calculate the exact type of objects allocated within the object or the typing context of generic method called within the object.”

That is, the type descriptors of Viroli are incompatible with and perform a differing function than those of claim 1. An inspection of the source code in Viroli on page 14, Figure 10 demonstrates that the fields of the typing context

Type of Response: Final Response
Application Number: 10/025,270
Attorney Docket Number: 180610.01
Filing Date: 12/18/2001

data structure contain translated objects and not the exact type. For example, on page 14, figure 10 of Viroli, the source code for the translated class "Client" has been translated to include a static collection of type descriptors (static \$TD[] \$t=new \$TD[4];) and the fields of the static collection of type descriptors "\$t" are then populated with calls to register types and calls to create the type descriptors for instantiations of the translated generic classes "Cell" and "Pair". The calls to create the type descriptors are in fact calls to the static "createTD" methods added by the translation of "Cell" and "Pair". More particularly, these "createTD" methods of translated "Cell" and "Pair" include calls to "\$TDManger.register" which store the instantiated translated objects in fields of the typing context data structure.

Therefore, during execution of the program the generic or "open type" class of Viroli may not determine the current typing context of the object as in claim 1 because the generic or "open type" class of Viroli has been translated and no longer represents the generic or "open type" class. In addition, as previously discussed and illustrated by Figure 10 of Viroli, the fields in the typing context data structure of Viroli do not describe the exact type; rather, the

fields in the typing context data structure of Viroli describe translated objects and not the exact type as in claim 1.

Accordingly, claim 1 is allowable over the cited reference and the rejection thereof should be withdrawn.

Claims 2-5 and 7-9 depend from claim 1 and are allowable at least by virtue of that dependency. Therefore, the rejection of these claims should be withdrawn.

Claim 25 has been amended to recite "an execution engine for executing parametrically polymorphic code and dynamically generating typing context data associated with a typing-context-relevant-code-point being executed within a typing context in a dynamic execution environment." The execution engine comprises:

(1) "a read module configured to encounter the typing-context-relevant-code-point in the typing context during execution of the program;"

(2) "a handle module configured to identify a typing context handle associated with the typing context, the typing context handle referencing a typing context data structure associated with the typing context;"

Type of Response: Final Response
Application Number: 10/025,270
Attorney Docket Number: 180610.01
Filing Date: 12/18/2001

(3) "a computation module configured to compute the typing context data associated with the typing-context-relevant-code-point;"

(4) "an allocation module configured to dynamically allocate a field in the typing context data structure associated with the typing-context-relevant-code-point, the field describing the exact type of the typing-context-relevant-code-point in the typing context;" and

(5) "a recording module configured to record the typing context data in the field of the typing context data structure." (Amendment emphasized).

Similar to claim 1, the field in the typing context data structure describe the exact type of the typing-context-relevant-code-point in the typing context. As previously discussed, Viroli makes use of translated objects and not an exact type.

Since each and every element of claim 25 is not disclosed or anticipated by the cited reference, claim 25 is allowable over Viroli and the Section 102 rejection of claim 25 should be withdrawn.

Claim 26 has been amended to recite "a method of dynamically generating typing context data associated with a typing-context-relevant-code-

point being executed within a typing context in a dynamic execution environment." The method comprises the steps of:

(1) "encountering the typing-context-relevant-code-point in the typing context during execution of the program;"

(2) "identifying a typing context handle associated with the typing context, the typing context handle referencing a typing context data structure associated with the typing context;"

(3) computing the typing context data associated with the typing-context-relevant-code-point;"

(4) "dynamically_allocating a field in the typing context data structure associated with the typing-context-relevant-code-point, the field describing the exact type of the typing-context-relevant-code-point in the typing context;" and

(5) "recording the typing context data in the field of the typing context data structure." (Amendment emphasized).

Similar to claims 1 and 25 discussed above, Viroli does not disclose a field in a typing context data structure describing the exact type of the typing-

context-relevant-code-point in the typing context. Accordingly, claim 26 is allowable over Viroli and the rejection of claim 26 should be withdrawn.

Claims 27-31 and 33 have been canceled, thus rendering the rejection thereof moot.

35 U.S.C. § 103 Rejections

Claims 6, and 10-14

Claims 6, and 10-14 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Viroli in view of U.S. Patent No. 5,093,914 issued to Coplien et al. (hereinafter "Coplien"). Applicant respectfully traverses the rejection.

Claims 6 and 10-14 depend from claim 1 and are allowable at least by virtue of that dependency for the reasons stated in the response to the rejection of claim 1, above. Combining Coplien with Viroli does not overcome the deficiency of Viroli that was previously described above, since Coplien does not teach or suggest dynamically allocating an exact field in a typing context data structure associated with the typing-context-relevant-code-point.

Accordingly, these claims are allowable over the cited combination of references for at least this reason and the rejection of these claims should be withdrawn.

Type of Response: Final Response
Application Number: 10/025,270
Attorney Docket Number: 180610.01
Filing Date: 12/18/2001

CONCLUSION

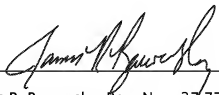
Accordingly, in view of the above amendment and remarks it is submitted that the claims are patentably distinct over the prior art and that all the rejections to the claims have been overcome. Reconsideration and reexamination of the above Application is requested. Based on the foregoing, Applicants respectfully requests that the pending claims be allowed, and that a timely Notice of Allowance be issued in this case. If the Examiner believes, after this amendment, that the application is not in condition for allowance, the Examiner is requested to call the Applicant's attorney at the telephone number listed below.

Type of Response: Final Response
Application Number: 10/025,270
Attorney Docket Number: 180610.01
Filing Date: 12/18/2001

If this response is not considered timely filed and if a request for an extension of time is otherwise absent, Applicants hereby request any necessary extension of time. If there is a fee occasioned by this response, including an extension fee that is not covered by an enclosed check please charge any deficiency to Deposit Account No. 50-0463.

Respectfully submitted,
Microsoft Corporation

Date: January 25, 2006
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399

By: 
James R. Banowsky, Reg. No.: 37,773
Attorney for Applicants
Direct telephone (425) 705-3539

CERTIFICATE OF MAILING OR TRANSMISSION [37 CFR 1.8(a)]

I hereby certify that this correspondence and the documents identified on this form are being electronically deposited with the USPTO via EFS-Web on the date shown below:

January 25, 2006
Date


Noemi Tovar

Type of Response: Final Response
Application Number: 10/025,270
Attorney Docket Number: 180610.01
Filing Date: 12/18/2001